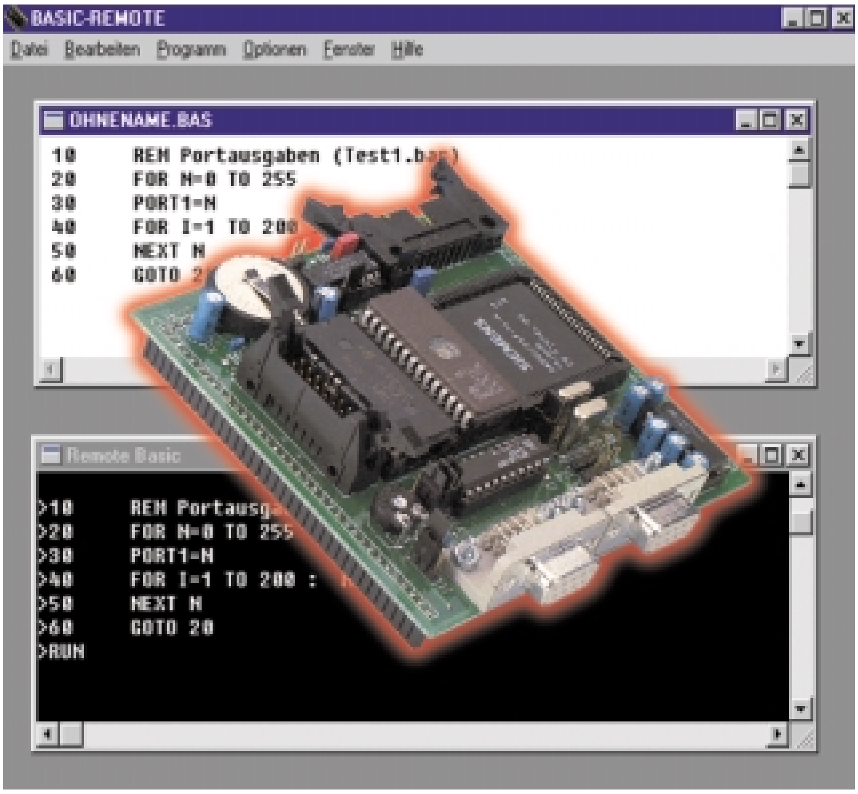
BASIC-537

a high-level language for the 80535/537

The most commonly used dialect of BASIC for microcontrollers, Intel's 8052AH-BASIC, cannot be used as is with the new 8051 derivatives. This article presents a modified version of the interpreter that also supports the 80535/537 microcontrollers. These have been used in recent Elektor projects, such as the 80C537 Single-Board Computer and the 537 'Lite' computer.

Design by B. Kainka



```
10      REM Portausgaben (Test1.ba
20      FOR N=0 TO 255
30      PORT1=N
40      FOR I=1 TO 20
50      NEXT N
60      GOTO 2
```

```
>10     REM Portausg
>20     FOR N=0 TO 255
>30     PORT1=N
>40     FOR I=1 TO 200 :
>50     NEXT N
>60     GOTO 20
>RUN
```

The Intel BASIC-52 interpreter was originally provided in the form of a mask-programmed microcontroller with the type designation 8052AH-BASIC. The interpreter automatically recognises the amount of available RAM and the baud rate of the connected terminal when the system starts up. There are also Autostart functions that allow a program to be run automatically on start-up. It is even possible to program an EPROM in the system, as long as you use the original Intel IC.

The 8052AH-BASIC IC is no longer produced. However, the program has been released for public use, so it can now be used legally in an EPROM. If you combine this with an 80C32 microprocessor, you have a low-power BASIC system. The only thing that you lose is the EPROM programming function of the original IC.

However, anyone who has tried to use a BASIC-52 interpreter in EPROM in an 80535 or 80537 system has experienced a bitter disappointment. The initialisation routine for the serial interface uses properties of the 8052 that have not been passed on to the 80535 in exactly the same form. The culprit is Timer 2, which is not compatible with the 8052.

The new processors can however be used if the reset routine is modified. The 8052-BASIC interpreter has provisions for a custom reset routine and for adding new user-developed instructions. We could thus write modifications that work with the new microcontrollers. In particular, the modified interpreter supports the Elektor 80537 Single Board Computer as well as the new 537 'Lite' computer.

80C537 Single Board Computer

- IC2 holds the BASIC-537 EPROM
- IC4 holds a 32-kB RAM
- IC5 and IC5 remain unused
- all 6 jumpers of J1A are connected to J1C (stand-alone mode)

537-Light computer

- J7:1—2 (12 MHz)
- J1A– J1C: 1—2 (stand-alone operation)
- switch S1: RUN

Table 1. 80537 computer board settings

The modified interpreter, which we call BASIC-537, has the following features:

9600 baud data transfer rate
with a 12 MHz clock
32kB RAM
6 new instructions
direct A/D converter polling
LCD support (address FFCOh)
Autostart function

The BASIC-537 interpreter occupies the EPROM memory region from 0000h to 403Fh, and also requires 32 kB of RAM starting at address 0000h. A few jumper settings must be changed on the 80537 boards, as shown in **Table 1**, to ensure that the address regions are properly configured.

In principle, any available terminal emulator program running at 9600 baud can be used with the BASIC-537 interpreter. After the system is switched on, the interpreter announces itself with the following start-up message:

```
MCS-51(tm) BASIC V1.1 / ES537
READY
>
```

You can now enter lines of BASIC code in the usual manner. Each character is sent back to the terminal via the serial interface. Each time a Return character is received, the interpreter converts the complete line into its internal token format and stores the tokens in the RAM. The program can be started by typing *RUN*, and it can be interrupted at any time by pressing *Ctrl-C*. Typing *LIST* sends the complete program listing to the terminal, and typing *NEW* clears the memory.

Instead of a simple terminal emulator, you can also use the Windows program BASIC.EXE, which is found on the same diskette as the interpreter. This is a combined terminal emulation and editor program that is specially designed for the BASIC-535 interpreter. The diskette also holds an instruction summary file (BASIC52.HLP), an initialisation file (SBASIC.INI), the binary file of the EPROM contents (BASIC537.BIN) and sample programs with the extension BAS.

THE PROGRAM EDITOR

When BASIC.EXE is first started, it is set up to use COM2. A different port can be selected via the *Options/RS232* menu. The selection is saved in the file SBASIC.INI when you quit the program.

When the microcontroller board is switched on, the BASIC-537 interpreter sends its start-up message. You can then enter the first program and run it. The sample program *Test1.bas* outputs data to port P1 and uses the original BASIC-52 instruction *PORT1*:

```
10REM Port outputs
20FOR N=0 TO 255
30PORT1=N
40FOR I=1 TO 200 : NEXT I
50NEXT N
60GOTO 20
RUN
```

BASIC.EXE has two text windows. A text editor with many editing functions runs in the upper window. The lower window displays all the characters received from the microcontroller. Every time that <Return> is pressed in the editor, the complete line is sent to the microcontroller.

The echoed characters appear in the terminal window. You can edit each line as much as you wish before it is transferred. This also applies to lines that appear earlier on in the text and that you want to modify after the fact. In the terminal window, you can see whether each line has been accepted or has caused an error message to be returned. Alternatively, you can activate the terminal window and use it for entering text, which will be transferred a character at a time.

The program's special functions can be called up using the *Program* menu. These are the *RUN, LIST, BREAK, CONTINUE* and *NEW* instructions for the interpreter in the microcontroller, and *CLS*, which clears the terminal window. You can use *LIST* to fetch a finished program and display it in the editor window, and then to save it to disk.

Naturally, instructions such as *NEW, LIST, RUN* and so on can be entered directly from the keyboard, the same as program lines. For testing, direct queries such as *PRINT PORT1* can be used. A special feature of the overall system is that code lines of a program that is already in the microcontroller are not overwritten by new lines unless the new lines have the same line numbers. This means that you can stock a collection of subroutines that can be downloaded as needed. In order to start from scratch, enter *NEW* before downloading a program.

You can load a program via the *File/Open* menu. First you should use *Program/Break* to stop any program that is already running, and then use *Program/New* to delete it. The text of the program will appear in a new editor window, and in the terminal window as well if the interpreter was ready. Several editor windows may be open simultaneously, which allows program sections to be copied back and forth.

INSTRUCTION SUMMARY

The most important instructions and functions of BASIC-537 are listed in **Table 2**. Numerous special functions have been adapted to the environment of the microcontroller. Some of the original BASIC-52 instructions cannot be used with the extended version and are not shown. Four new instructions have been specially added for the 80537.

The measurement range of the twelve possible analogue inputs is 0 to +5 V, but this can be modified within wide limits using the instruction *DAPR*, which writes a parameter byte to the processor register with the same name. A one-byte control parameter can be specified for each measurement. Its two 4-bit nibbles set the upper and lower limits of the range. Bits 0–3 set the lower limit in units of (5 V/16), and bits 4–7 set the upper limit in the same way. These two reference voltages must have a minimum separation of 1.25 V. For example, a parameter value of 84h sets the measurement range to 1.25–2.5 V. The value 00h is an exception; it sets the range to 0–5 V.

The instructions *SFR* and *WrSFR* allow all of the special hardware functions of the 80C537 to be used. They provide access to both serial ports, the timer, all supplementary ports and so on. Providing free access to all special-function registers presents a special challenge to an interpreter, since all 8051 derivative ICs allow only direct addressing (via an instruction such as `mov 90h, A`). The *SFR* address thus cannot be passed via another register. A compiler has no problem with this, since it hard codes the address, but an interpreter must be able to pass the address. This difficulty is probably the reason why the original BASIC-52 interpreter allowed free access to the internal RAM, the external RAM and even the program region, but not to the special function registers. Since a 32-kB EPROM has enough room for the necessary extensions, an indirect approach can be used. There are two small subroutines for reading and writing every address in the range from 128 to 255. When the interpreter encounters an *SFR* or *WrSFR* instruction, it calculates a jump address that leads to the proper subroutine. The extended interpreter is thus prepared for microcontrollers that haven't even been invented yet.

The text file *Sample Programs*, which is also located on the diskette, contains programs that illustrate the use of the specific instructions of the interpreter.

LCD CONTROL

BASIC-52 allows serial interface outputs to be redirected to a different output routine. This capability is used in the BASIC-537 interpreter to drive a liquid-crystal display. The display is largely addressed using normal instructions. Only the instruction for initialising the LCD (*LCDINIT*) and the instruction for setting the character location (*CURSOR*) have been added.

The new 537 'Lite' computer (January 2000) has an interface for an LCD module. This is located in the address space starting with address FFCOh. The display is initialised using the new instruction *LCDINIT*. After this, there are several ways to output characters

Instructions

RUN Ctrl-C CONT LIST NEW

Operators

*+ - / * = > >= < <= <> .AND. .OR. .XOR. ABS() NOT() INT() SGN() SQR() RND LOG() EXP() SIN() COS() TAN() ATN()*

Statements

CALL DATA READ RESTORE DIM DO-WHILE DO-UNTIL END FOR-TO-STEP NEXT GOSUB ON-GOTO ON-GOSUB IF-THEN-ELSE INPUT LET ONERR PRINT REM STOP

Extended BASIC-537 Statements

ONEX1	*subroutine call following Interrupt 1*
ONTIME	*timer interrupt call*
RETI	*end of an interrupt subroutine*
PH0., PH1.	*output a hexadecimal number with/without leading zeros*
PUSH, POP	*move data to/from the argument stack*
STRING	*reserve memory for a text string*
IDLE	*wait for interrupt*

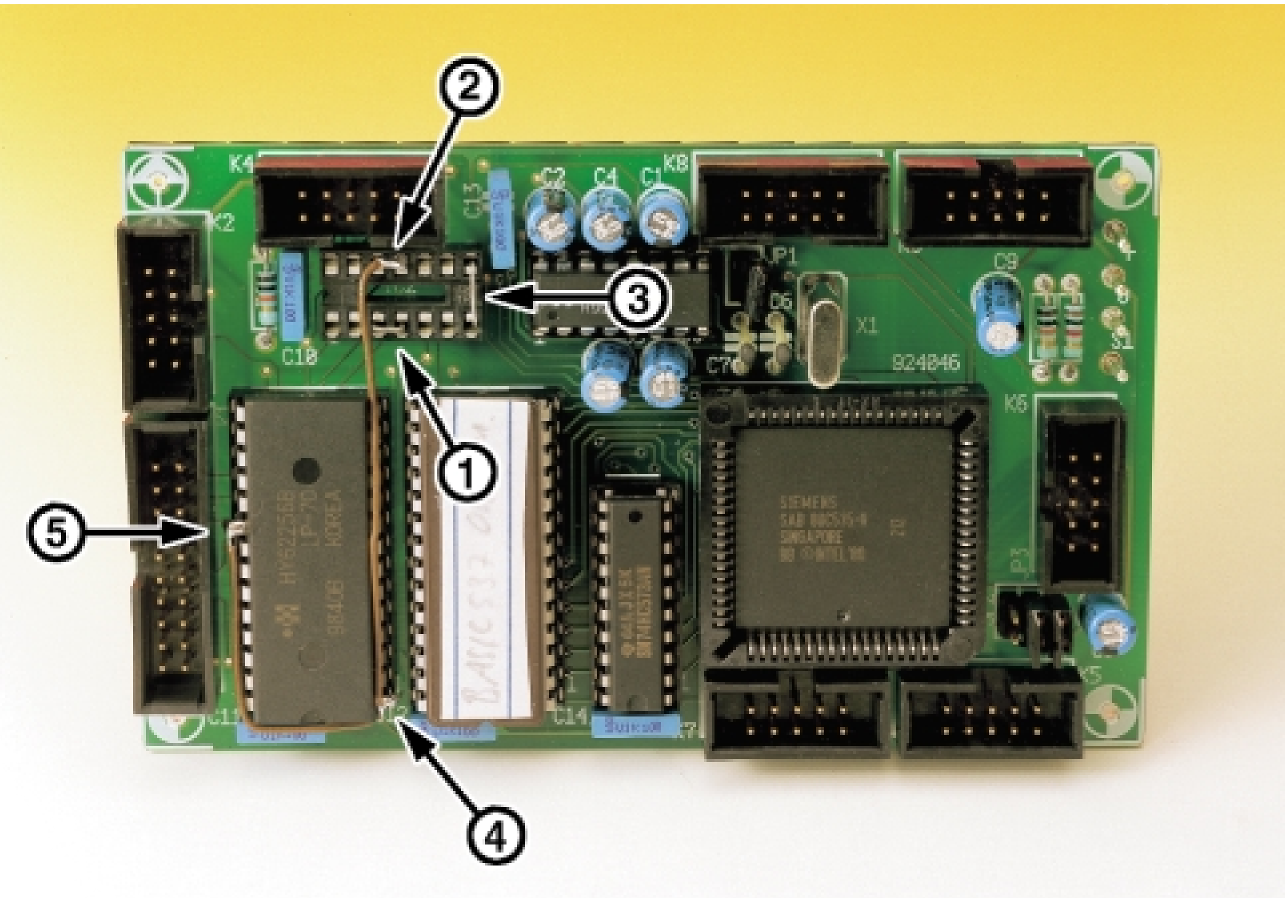
Special function operators and system variables

CBY() DBY() XBY() GET IE IP PCON T2CON TCON TIME TIMER0 TIMER2 PI XTAL MTOP LEN FREE

80C537 I/O handling

AD (channel)	*[read-only]*	*C = AD (0)*	*analogue input (0 – 5V) at the AD0 input*
		C = AD (11)	*analogue input (0 – 5V) at the AD11 input*
DAPR	*[write-only]*	*DAPR = 128*	*switch the measurement range to 2.5 V*
SFR	*[read-only]*	*print SFR (90H)*	*reading special function registers*
WrSFR	*[write-only]*	*WrSFR 90H,255*	*writing special function registers*

Table 2. Summary of BASIC-537 instructions



to the display. The desired character position can be specified using the *CURSOR* instruction. The *PRINT@* instruction writes directly to the display. A sample program is available on the diskette or from the Elektor website.

A program can send its outputs to the terminal emulator using *PRINT*, or to the LCD using *PRINT@*, as desired. All output data can be formatted with the *USING* instruction. However, the line feed function has no effect on the LCD. This means that the CURSOR instruction should be used as much as possible to specify the desired location of the output data. Various types of LCDs can be used, with one, two or four lines. The sample program *LCD.BAS* shows the character position corresponding to the start of each line.

Another option is use the instruction *UO 1* to redirect the entire output to the LCD. The instruction *UO 0* switches the output back to the terminal

AUTOSTART

The original BASIC-52 tests the entire RAM after start-up, clears it and then starts any program that it recognises in the external EPROM. Both of the Elektor 80C537 computers have battery-backed RAM, so that the program that was last run before the computer was switched off remains present in the RAM. The BASIC initialisation routine has therefore been modified to automatically start a program that is already in the RAM.

All that you have to do is to load a program into the computer's memory. After this you can disconnect the serial cable to the PC and let the computer work on its own. Each time the computer is started up, the program is run anew, at least as long as the lithium battery provides enough power to maintain the contents of the RAM. If at any time you don't want the system to automatically run a program when it starts up, all you have to do is to send it a *NEW* instruction.

WE HAVEN'T FORGOTTEN THE 80535!

Now we come to the widely used 80535 systems, such as the Elektor 80535 Single Board Computer. The new interpreter can also be used with this system, if you first carry out a few modifications.

The system must run at 12 MHz, in order to yield a data transfer rate of 9600 baud. Everything else works as well, as long as you take the available processor hardware into account. The A/D converter has only eight channels, but is otherwise compatible. Naturally, the 80535 has fewer timers, ports and so on than the 80537.

Intel's BASIC-52 interpreter expects to find data RAM starting at address 0000h. The new initialisation routine does not change this. However, the 80535 Single Board Computer divides its 32 kB of RAM between the address ranges 4000h–7FFFh and C000h–FFFFh. Special logic allows the RAM in these regions to be used to store program code as well as data.

The necessary changes are quite simple. Remove IC2 (a quad NAND IC, type 74HC00) from its socket and replace it with three small wire bridges, as follows:

pin 3 to pin 4	($\overline{OE}$ direct to $\overline{RD}$)
pin 11 to pin 12	($\overline{CE}$ direct to *A14*)
pin 7 to pin 8	($\overline{CS}$ direct to ground)

In this configuration there is 16 kB of RAM available to the system, starting at address 0000h. The normal BASIC-537 interpreter can work with this. Of course, you must tell the interpreter that only 16 kB of RAM is available, by entering:

```
MTOP = 16383
```

This line should also be placed at the beginning of every program, since the interpreter initialises itself to 32 kB of RAM after a restart. Since all variables are stored just below the upper RAM boundary, the interpreter will function correctly only if it has accurate information about the amount of available RAM.

If you would rather be able to use the full 32 kB of RAM, you must make two further modifications, as follows:

Bend pin 1 of the RAM IC (A14) so that it does not stick into the socket, and then connect it directly to address line *A14* (for example, at pin 12 of the socket for IC2).

Bend pin 20 of the RAM IC ($\overline{CE}$) so that it also does not stick into the socket, and then connect it directly to address line *A15*, which is on pin 1 of the same socket.

Now the system will see the full 32 kB of RAM in one continuous block, starting at address 0000h. With this modification, it is not necessary to specify the amount of available memory with *MTOP*.

The 80535 Single Board Computer does not have battery-backed RAM. However, it is possible to use a 'zero-power' RAM, which has a built-in backup battery. If such a RAM is used, this computer behaves the same way as the new 80537 computer. With a zero-power RAM you can have a program start automatically. Even without this luxury, though, the Autostart function of the extended BASIC interpreter makes its presence known. This can be seen if a reset is executed while power is still applied. In this case the program that is already present starts immediately after the reset. This is a quite useful feature during program development. Instead of entering *Break* and then *Run*, you can simply press the Reset button.

An additional difference can be seen in the fact that it is possible to use a liquid crystal display. The '535 computer does not have any decoding logic for the LCD, so you will have to either build some extra circuitry or else connect the LCD to a port and use the 4-bit mode. The display can be driven by a somewhat complicated BASIC routine.

All of the supplementary ports of the 80535 can be addressed not only via their SFR addresses but also directly. The instructions *P3*, *P4* and *P5* can be used for output only. An output value can be directly assigned to a port (e.g. *P4* = 255). Reading input data from the ports with the *InP3*, *InP4*, *InP5* and *InP6* instructions is unfortunately not as convenient, since this must be done via the result stack. Port states that have been read can be fetched and put into a variable using the POP instruction:

```
InP4 : POP A : PRINT A
```

The direct port instructions also work with the 80C537, but only up to port P6. BASIC-52 allows only a limited number of new keywords. It is thus not possible to give every new function its own name. This means that we have to continue programming the special hardware of the microcontroller via register addresses. With read operations in particular, the use of register addresses (e.g. *Print SFR(0E8h)*) is often more elegant than the use of direct port instructions.

It is relatively easy to write some reset routines and supplementary instructions yourself, since the BASIC-52 interpreter already has suitable interfaces. You can find more detailed information in the relevant literature.

(000018-1)

You can find additional information and sample programs for 80535 and 80537 microcontrollers, and a DOS terminal emulator for BASIC-52, at the author's homepage on the Internet: http://home.t-online.de/home/B.Kainka

Project Software

Diskette, order code **996029-1**.
(terminal emulator, help file and sample programs)

EPROM, order code **996532-1**.
(BASIC-537 interpreter)